Venture Cover

issue 19.

AMSTRAD ADVENTURE P.D.

Don't let your adventures collect dust - Get your work seen!!

A newly formed PD specialising in adventures only is seeking home-brew adventures to include in the collection. Send your tape or disc to:

ADVENTURE PD, 10 OVERTON ROAD, ABBEY WOOD, LONDON, SE2 9SD

For more information send a SAE to the above address.

E is for EDITORIAL

Regular readers will have noted a distinct gap of quite a while between this and the last issue! It's apologies time again - sorry! But the truth of the matter is that I've been busy getting to grips with my new computer and printer! This is the first Coder to use this setup, and I'm not sure just how it will look printed. Hopefully there'll be some more improvements next issue.

If you've anything to say about the magazine - be it positive or negative - don't hesitate to write in and let me know. And don't forget I'm still looking for contributions to 'Your Say' - that's the page entirely open to any reader to say what they want - about anything!

The last issue of Coder had a free Spectrum adventure with it, and this issue it's the turn of Commodore 64 disk users! On page 7 is a coupon enabling you to send off for a free disk, crammed with programs! These are listed on page 9. I decided not to include a free disk with each magazine as I know there aren't as many of you own Commodore 64s as there are Spectrum owners. Plus you might not also have a disk drive, hence the coupon. And yet again, this freebie comes to you without any increase in the price of the magazine! (And Coder is still the cheapest of the adventure fanzines!) I hope you all appreciate this generosity - maybe I should have charged an extra pound!

The good news is... there'll be another free disk next issue as well! Complete with a whole adventure game to play! (Ex-Editors permitting! Ahem.)

I hope you enjoy this new issue then, and once again, I must apologise for it being so late!

Yours,

The Editor.

P.S. Illness has delayed this another week! I was too ill to paste it up last week!

So the lemon said to the banana, "How about some letters then?"

"What a good idea," replied the banana, "but don't tell the apple."

Robin Rawson-Tetley writes...

I bought your magazine upon seeing it mentioned in 'Sinclair User', as I love writing adventures (I am also a masochist) and I was extremely impressed. What particularly impressed me was Stephen Groves' guide to writing an adventure system. I am currently writing my own, and thought I'd offer a quick (ish!) tip: Turn to Issue 18 - Stephen rightly says in regards to memory that "every little counts", however, I spotted a quick way to save 8 bytes from looking at the listing. Cast your attention to pages 23 and 24, look at all those "JP ARROW" instructions, alter these to "JR ARROW" and you've saved 8 bytes! The reason is that a JP instruction loads the program counter with the values held in the 2 adjacent memory locations (the physical memory address where "ARROW" starts), it therefore takes 3 bytes - 1 for the JP opcode and lo and hi bytes. A JR instruction calculates the distance between the JR and the start of ARROW and puts it in the next byte - taking only 2 bytes, however this 'offset' is added to PC+2 and uses signed arithmetic, so it cannot be used in cases where the jump to be made is greater than either -126 or +129 bytes. But since ARROW was far less than 129 bytes ahead, a JR made sense. (In cases where a jump is out of range, a JP must be used.)

Therefore every replacement of a JP with a JR saves one byte. The only situation where the jump is in range and a JP is used is the fact that JP is 2 clock cycles faster than JR - a measure of time that is indescribably insignificant.

Also, I believe that on page 21, a "BORD6" label should have been placed next to the "PUSH BC" instruction, 12 lines down. I apologize for the length of this letter, but I hope I've helped some people (and not upset Stephen, his 'Thingy' is brilliant) and I think your mag's great. (I enjoyed 'Pride Of The Federation' also, and have a quick tip: Can't examine anything? Use EXAM instead!)

Jim Read writes...

Just a few lines to thank you for the 'Pride Of The Federation' adventure together with the A.C.; I have not yet completed the adventure but keep trying.

Martin Bela writes...

I agree with most of the points made by Steve Clay concerning adventure standards (see A.C. Issue 16 p.8); however, his comments about highlighting token words during conversations would seem to suggest that adventure players are somewhat dim. After all, isn't that half the fun; finding out what a character will talk about? The programmer should of course ensure that replies from characters give a clue as to what to talk about.

Spelling is another problem. A few minutes spent with a dictionary would make many good games even better. I wonder how many games have *no* spelling mistakes? I am writing my first adventure (on a CPC6128) using the Protext and ADLAN ROMs. This means that I can let the Prospell ROM check all

my spelling. I've not used any other adventure creator so I don't know how easy it is to check the spelling of GACed or PAWed games, but a spelling dictionary is fairly quick to use. The only problem is that (if today's educational standards are anything to go by) a lot of the writers will need to check everything over four letters long!

Similarly, few adventures have the correct style and presentation. For instance...

"Yes," he replies, "We have some rope."

...is incorrect; the split quotation has a capital letter in the middle. The correct way is...

"Yes," he replies, "we have some rope."

Another point of confusion is the correct placing of an apostrophe. If you wish to indicate the possessive case of a singular noun, then the apostrophe should go before the last letter, eg:-

The farmer's wife gives you some food.

If you're indicating the possessive case of a plural noun, then the apostrophe goes after the last letter, eg:-

The two farmers' wives talk amongst themselves.

All clear? ...I hope so.

The single most important feature of an adventure game is character interaction. Adventures can be very boring if there's no-one to talk to; so put plenty of characters in your games. If you're including graphics then do some pictures of your characters. It's nice to actually see the person you're talking to.

I think that too many people place too much importance on the complexity of the parser. After all, is it really necessary for the computer to be able to understand a single input of ten or twenty words? I think not! A parser that understands two or three word inputs is usually all that's necessary. It is important however, that the parser has a comprehensive vocabulary; so that the player does not have to spend a lot of time finding exactly the right verb to use.

Why don't writers pay more attention to the finishing touches? The game I'm writing at the moment has two text windows: one for the player's input, and one for the descriptions etc. These windows are bordered by suitable graphics, such as stone pillars, vines, and snakes. Does anyone else do this? *(Ed - my last GACed game had graphics bordered by a frame of leaves, squirrels and flowers!)* The nearest I've seen is the bit of border used in 'Lord Of The Rings'. Again, perhaps this isn't possible with most adventure creators, but you can with ADLAN, and of course with BASIC.

In conclusioon then, I guess that the standard of adventures will follow that of education; computer programming skills will improve (no doubt making parsers even more complex), and basic English grammar skills will decline, leading to wurdz spelt wrong, apostroph'es in the

wrong place, and capital letters where they shouldn't Be!

Steve Clay writes...

I must agree with Tom Frost that there is a difference between using PAW and actually programming an adventure. Gilsoft have produced a product of immense excellence, indeed most adventures written with PAW are superior to other non-utility products. The response time of PAW makes playing Level 9 games very trying. The difficulties involved in using machine code (m/c), let's face it BASIC is too slow, are vast. You have to worry about everything, from clearing the screen to printing a full stop. The registers need constant watch. I have used m/c and I know how awkward it can be. One mistake in m/c normally (99 times out of 100) means the computer locks up. A reset and a reload follow. A mistake with PAW just means that the puzzle doesn't work or the wrong message is printed. Gilsoft have made our lives much easier.

LINE ADS

ADVENTURES WANTED FOR PLAYTESTING! Commodore 64 tape/disk only. Robin Rawson-Tetley, 33 Furnival Way, Whiston, Rotherham, South Yorkshire, S60 4BQ.

C64 GAMES FOR SALE! Only 50p per tape, £1 per disk. TAPES: Pitstop II, Everyone's A Wally, Revenge Of The Mutant Camels, Bounty Bob Strikes Back, Rockford/Back To Reality, Anarchy, Tetris, Suicide Express, Pyjamarama, Pastfinder, Antiriad, I Ball, I Ball II, Gogo The Ghost, The Trap Door, Gauntlet, ZZAP! Sampler 2 (Cybernoid 2, Hawkeye), ZZAP! Sampler 3 (Thunderblade, Mad Mix) ZZAP! Sampler 4 (Robocop, Parallax), ZZAP! Sampler 5 (Phobia, Zig Zag, Music), ZZAP! Sampler 6 (Sanxion, Mutants), ZZAP! Sampler 7 (Oh No!, Dominion), ZZAP! Sampler 8 (Galax-I-Birds, Felix), ZZAP! Sampler 9 (Starace, Dicky's Diamonds), ZZAP! Sampler 10 (Foxx Fights Back, Creatures screen, Scorpion, Monster Munch), Commodore User tape (Operation Wolf, Exceleron), C+VG tape (Brainstorm), PCG Christmas Gift tape (12 demos!), Commodore User tape (Hyperactive, Dominator, 720°, Outrun), Radio ACE (music tape to listen to), Frankie Goes To Hollywood. DISKS: The Sentinel, Mercenary - The Second City, They Sold A Million (Daley Thompson's Decathlon, Beach Head, Staff Of Karnath, Jet Set Willy), Paradroid/Uridium Plus, Driller, Rainbow Islands, 4th Dimension (Cyberdyne Warrior, Insects In Space, Head The Ball, Mission Impossibubble), Wizball, ZZAP! Sizzlers II (Z, Monty On The Run, Starquake, Bounder), Quedex, Intensity, Bombuzal, Hunter's Moon, GEMS - 3 arcade games, 1 adventure, Hercules/Gods & Heroes, Elite, Red L.E.D., Bubble Bobble, Beamrider, The Image System, Captain Blood, Spindizzy, Boulderdash/Rockford's Riot (only disk copy ever released!!). Chris Hester, 3 West Lane, Baildon, West Yorks., BD17 5HD.

YOUR SAY: George March

The page that YOU the reader writes... about what you like!

In answer to Roy Milliken's quoting me in Coder Issue 16, on a letter I had printed in Issue 15, where he mentions that he has "Nothing but doubt over the phrase followed by the double ?? in the letter of George March in Issue 15. 'Heads Cases' ?? Well, that is hellish grammar for a start." The phrase that Roy mentions was in fact (Okay, I've split the word up for you exactists out there!)... "God knows what he (Ian Eveleigh) thinks of 'Heads Cases' speech??" Which actually should have been 'Head's Cases', and I was in reality talking of *Richie Head's* dialogue used by me in the 'Head's Cases' (Ie, the Cases belonging to Private Investigator Richie Head) stories I've written for both 'Adventure Coder' and 'Workshop', most of the abbreviations of which come from the G & C Merriam Co' & Longman's Dictionaries, 'Americanadian Slang and 'breviations diction'ry' (Not you will notice 'Abbreviations Dictionary'! Their title not mine!). It's supposed to be a funny view of various American/Canadian accents and selections of everyday dialogue!

My last point on this 'Communications Breakdown' controversy that has occured in recent issues of Coder is the NEED for a basic 'individuality' of prose and accents in communication, and if I can again quote Roy Milliken from Issue 16, where he says "...If we are to have a *sensible* kind of society, then we must accept some form of *standards*." But, each different style belonging to any area of the British Isles, not only in speech, but in written communication also, gives each area, or separate dialect, its own 'individual' personality, and taking away that difference between each area/dialect by 'standardising' the English language logically takes away each area/separate dialect or accent's *individuality* and *personality*!

When my English Language lecturer at the New College Newcastle (it used to be the College of Arts and

---

using basic character blocks for the 3D designs. (Note how I copied the score panel from the Amstrad version of the game!)

6) Jazz - a portrait of Louis Armstrong complete with animated title - leave it running for a while! (Also one of my VERY earliest screens - ahem!)

Boulderdash Kit games: there are two to play, for all owners of the 'Kit', who need to load that up to use these game-files I'm afraid. They're guaranteed addictive!

"ROCKY.GAM" is the first game, which allows you to start from every fourth screen, just like the original game of Boulderdash. But if that proves too hard, I've slipped on "ROCKY2.GAM" for any cheats who wish to start from any screen! "Rocky" consists of 15 screens to play in the normal manner (eg: get the diamonds before the time runs out, move the rocks, kill the fireflies, make more diamonds with the butterflies and the magic walls, get the Pounds

Technology, C.A.T.) started his first lesson for us gremlins, he played us a tape of the 'Little Water', Bobby Thompson, as an exercise in demonstrating how an *individual* dialect (Geordie in this case) can be used to great advantage in communications styles, ie, some of the funniest comedians use their *individuality* of language styles in their stage, or TV acts. For example Arthur Mullard (can you imagine him doing Bill Shakespeare's "To be or not to be"?), Professor Stanley Unwin's 'Gobbledygook' langauge, Bobby Thompson, Bob (Roger Rabbit) Hoskins, Master Chef/Through The Keyhole's Lloyd Grossman, Bread's Ma' Boswell, the Liver Birds (anybody remember them?), comedian Stan Boardman (would they still be the same if they didn't have Liverpudlian accents?), would TV's Byker Grove still be Byker Grove without the accents? (Even though the Grove is/was mostly filmed in and around the Scotswood and Lower Benwell areas of Newcastle and NOT Byker!)

Now somebody's bound to say "But accents are not the written language, if you standardise the written word then individual dialects and accents will still be the same!". I don't know? And my lecturer agrees, if you standardise the English language (which according to him isn't really one language but a 'bastardisation' of the original Pictish/Celtic, with Roman, Swedish, American, Portuguese, and God knows what else added!), then somebody will want to standardise every other language, 'bastandardised' American, Greek, Russian or whatever, because these foreign words that we now use have become assimilated into our own everyday dialects (I mean how many people actually realize that the word 'shampoo' is actually an Indian word?), and if you standardise any foreign words used in English then you are actually standardising parts of someone else's language!

I've come to the conclusion that standardising English is a losing battle, as new words (not only English but foreign as well!) are being added to the language every day!

George March.

(Foxx Fights Back, Creatures screen, Scorpion, Monster Munch), Commodore User tape (Operation Wolf, Exceleron), C+VG tape (Brainstorm), PCG Christmas Gift tape (12 demos!), Commodore User tape (Hyperactive, Dominator, 720°, Outrun), Radio ACE (music tape to listen to), Frankie Goes To Hollywood. DISKS: The Sentinel, Mercenary - The Second City, They Sold A Million (Daley Thompson's Decathlon, Beach Head, Staff Of Karnath, Jet Set Willy), Paradroid/Uridium Plus, Driller, Rainbow Islands, 4th Dimension (Cyberdyne Warrior, Insects In Space, Head The Ball, Mission Impossibubble), Wizball, ZZAP! Sizzlers II (Z, Monty On The Run, Starquake, Bounder), Quedex, Intensity, Bombuzal, Hunter's Moon, GEMS - 3 arcade games, 1 adventure, Hercules/Gods & Heroes, Elite, Red L.E.D., Bubble Bobble, Beamrider, The Image System, Captain Blood, Spindizzy, Boulderdash/Rockford's Riot (only disk copy ever released!!). Chris Hester, 3 West Lane, Baildon, West Yorks., BD17 5HD.

WHAT'S ON THE FREE C64 DISK THEN, JOHN?

STAR LC-10 NLQ Font Designer: a program to design your very own printable type! Don't be stuck by the normal ones, create your own! Whole alphabets, symbols or scientific characters, whatever you want that the Star can't do normally! Comes complete with two free ready alphabets and program to download the designs into the Star! (Load "INFO" for full instructions first!)

Graphic Demos: a fully linked suite of demos for you to view! Complete with 'GEM'-like interface (windows and arrow!) allowing access to each demo, even the disk directory and error channel! Choose a demo, or start at the first one and link from there to the end! But a note of warning: these demos aren't exactly technical masterpieces packed with raster and scrolling fx but a collection of ideas based on games. Mostly written in BASIC, I did pop in some machine code to speed up the sprites! Also, I did these years ago, so they're far from state of the art! But definitely worth a goggle. The demos are:

1) Cave - move the sprite across three cave screens. Note the awful speed of this all-BASIC affair! But also the colourful sprite designs and animations. (It's a bit crap really this one!)

2) Robots - a scene from a possible game? There are several floors of an office complex on screen, which have been taken over by robots, who patrol them at rapid speed. That's it, really!

3) Iridium - a blatant rip-off of a shoot-em-up with a similar name. You can't shoot anything unfortunately, but aren't the sprites good?

4) Wizard - this is a picture of the old ZZAP! White Wizard I did, complete with touches of animation. Watch the window and the face!

5) Spin - a screen to show how Spindizzy was put together using basic character blocks for the 3D designs. (Note how I copied the score panel from the Amstrad version of the game!)

6) Jazz - a portrait of Louis Armstrong complete with animated title - leave it running for a while! (Also one of my VERY earliest screens - ahem!)

Boulderdash Kit games: there are two to play, for all owners of the 'Kit', who need to load that up to use these game-files I'm afraid. They're guaranteed addictive!

"ROCKY.GAM" is the first game, which allows you to start from every fourth screen, just like the original game of Boulderdash. But if that proves too hard, I've slipped on "ROCKY2.GAM" for any cheats who wish to start from any screen! "Rocky" consists of 15 screens to play in the normal manner (eg: get the diamonds before the time runs out, move the rocks, kill the fireflies, make more diamonds with the butterflies and the magic walls, get the Pounds

for extra points, but avoid the Deutschmarks - the usual stuff.) Here's a rundown on each screen then:

1) WELCOME - can you sort out the maze, avoiding the nasties?

2) TRAPDOOR - watch out for the expanding walls!

3) ENCLOSED (interval) - like all the intervals in the game, just go for those diamonds as fast as you can! But how?

4) 99 - once you've sussed things, it's quite easy really!

5) GARDEN - where are all the diamonds you need!? Have fun!

6) COLUMNS (interval) - a bit tricky?

7) FUN2 - one of my best designs - it's in the shape of a house! But can you make it through each room to the attic? (Sounds like 'Jet Set Willy'!)

8) METRO - futuristic frolics! You'll need to think here.

9) VAULTS (interval) - tough!

10) UPANDOWN - keep moving!!!!

11) CHEM3 - short for Chemistry, again a futuristic one.

12) GATEPOST (interval) - clever idea, eh?

13) MIDNIGHT - my real shocker. You won't believe this one but... THE SCREEN'S BLANK!!! So you have to use your EARS to get all the diamonds, but believe me, it CAN be done!

14) TOMB - very tough. Exact movements are required to avoid the fireflies!

15) PROTECT (interval) - keep those nasties from the Rockfords!!

This set of caves just goes to show what fun you can have making your own Boulderdash screens! With little effort at all you can soon build up the most amazing designs from the simple elements available - the Kit program is brilliant!!

"JANE.GAM" - this is my second (brief) game, made up of ideas left over from the first game.

1) CALAMITY (get it?) - this is a killer! No I don't mean hard, but watch what happens when you first enter the cave!! I had IMMENSE fun setting this one up!!

Then there are three simple interval caves to finish with:

2) ROWS (interval)

3) MAZETTE (interval)

4) TEA (interval)

From 'Bare-Bones' to 'Goblin Gazzette'...

...The Editor investigates.

I was planning a review of 'Bare-Bones' #4, when 'Goblin Gazzette' #5 arrived - they're the same magazine! Les Mitchell has renamed Bare-Bones, and the spelling of Goblin Gazzette is deliberate - not a spelling mistake! (Blame John Wilson - he thought it up!) Les felt the mag needed a new name because he's recently been expanding it to cover not just adventure solutions, but the whole playing scene, with reviews, tips, competitions, readers' letters, and more!

Issue 5 is a big 37 pages long (which must be impossible - folded sheets go in multiples of 4!) and seems to be good value at £1.50. There's a review of 'Aura-scope' for the 128K Spectrum, 'Double Agent' for the Amstrad PCW 6512, 'The Challenge' for the CBM64, 'Colditz Escape' for the Amstrad CPC 464/6128, and a preview of 'Dragon Slayer' for the Spectrum too; so it's clear there's a wide range of machines covered.

The print quality is excellent, and text is easy to read. Plus a selection of the 'usual' homegrown adverts - why don't I ever get these!?!?

The letters pages are just like those of 'Adventure Probe' - lively, and full of debate.

Overall, my impression was of a great 'all-round' adventure magazine. I can't think of anything bad to say about it, so I won't! Indeed, it's very close to the format and user-friendly nature of Probe. Only it's 50p cheaper! So my advice is to forget Probe and get Goblin Gazzette instead! Though I doubt many would agree with me there! Oh well, just an idea! All credit must go to Les for producing such a magazine of such obvious high standard, and here's to its future!

WRITE To:

Les Mitchell

10 Tavistock Street Newland Avenue Hull HU5 2LJ

HANDY TIPS: Star LC-10

by Christopher Hester

The Star LC-10 is a 9-pin dot matrix printer. It offers four resident fonts (ie: type styles) that you can use when printing in Near Letter Quality Mode, or 'NLQ' for short. When you first switch on the printer, it's working in Draft Mode, and you need to select NLQ Mode to use the four fonts. The reason is that in NLQ Mode, the Star prints each line twice, which enables it to print over each letter with more dots of ink, giving twice as much detail, or quality, as you get in Draft Mode, where a line of text is only printed once. Draft Mode just doesn't allow enough detail for several styles of text.

Don't just stick to the first of the NLQ fonts - try each one. Font No.1, 'Courier', is like a standard 'typewriter' font, and the second NLQ font 'Sanserif' looks to me like the style you get from those early Amstrad printers, with flat-shaped letters. Whilst 'Courier' might be best used for an 'official' letter, to a bank, or a business, 'Sanserif' would appear to be the font you'd use for friendly letters.

The third and fourth fonts available are 'Orator', in two versions. The first uses capital letters only, but with smaller ones for lower case letters. This is odd, but in practice can look quite professional. Don't dismiss it. The second variant of 'Orator' uses lower case letters, but they're larger than usual, giving an 'American', or 'Modern' feel to it. The drawback with that one is that letters like 'g' and 'f' have had to be squashed to fit the larger size. This is because the printer only has so much space on its print head, that it uses to print each character of text on the page. Nine pins of space, to be exact. Each letter is in fact made up using just these 9-pins, which are a bit like 'pixels' on your computer screen. More costly printers have a 24-pin print head, so they can produce not larger, but more detailed letters. The more pins a printer has, the better the output it can produce. Yet despite only having 9 pins, the Star LC-10 has been so well designed that I was once fooled by a print out I saw from one - I was convinced it was off a 24-pin printer, it was so good! The 'Orator' font is a superb testimony to this, in that the font almost looks too good for a 'mere' 9-pin printer!

Here are examples of each font, so you can see the differences:

This is FONT No.1 - 'Courier'

This is FONT No.2 - 'Sanserif'

THIS IS FONT No.3 - 'ORATOR 1'

This is FONT No.4 - 'Orator 2'

You might think that was all there was to it, but these four simple fonts can offer much more variety than you think! You can also use them in a wide range of additional modes. The Star LC-10 offers several of these; in Elite Mode, each letter is slightly compressed, giving the result below. You can use this mode to fit more letters on a line.

This is FONT No.1 - 'Courier'

This is FONT No.2 - 'Sanserif'

THIS IS FONT No.3 - 'ORATOR 1'

This is FONT No.4 - 'Orator 2'

Then there's Condensed Mode, which is actually what I'm using for this text here. That's to make it easier for copying in the magazine, but normally this mode is too small to be of much use. Certainly it's not advisable for long letters! But look at the same fonts again using Condensed mode, and see how much they've changed!

This is FONT No.1 - 'Courier'

This is FONT No.2 - 'Sanserif'

THIS IS FONT No.3 - 'ORATOR 1'

This is FONT No.4 - 'Orator 2'

You can now see that I'm using 'Orator 2' for the main text here. 'Courier' is not too good in Condensed Mode because of the way it uses 'serifs' - those are the little ledges around each character that end each part of the letters. 'Sanserif' mode doesn't have these, hence its name 'Sanserif', from the French 'sans serif', meaning ''without ledges'', see?

One use of Condensed Mode is to create text small enough to fit under a picture and give a caption. Or you could use it for a large table of figures where you need to fit text in there as well.

The next mode that's highly useful is Expanded Mode. This is the opposite of Condensed Mode, and expands each character being printed to twice its original size. Here are the four fonts in Expanded Mode:

This is FONT No.1

This is FONT No.2

THIS IS FONT No.3

This is FONT No.4

Note how the ~~horizontal~~ Vertical lines of each letter have become thicker.

In the Star manual, it also lists two more useful modes - Emphasized Mode and Double-strike Mode. The first thickens the VERTICAL lines of each letter, resulting in bolder text, whilst Double-strike Mode thickens the HORIZONTAL lines instead, again to give you a bolder looking text when you print. If you select the two modes at once, you'd expect both the horizontal AND the vertical lines of your text to be thickened. This does indeed happen - but only in Draft Mode! If you've selected NLQ Mode instead, you'll find that Emphasized Mode has no effect on your text! This is

odd, as if it works in Draft, so why not NLQ?

You might think that was the end of it - not so! There IS a way to achieve vertical thickening of NLQ letters! But you won't find it in the Star manual! The point of thickening the text in the first place, is to make it bolder, and darker. If you're using a weak ribbon, this can be crucial to making your text more readable! In Draft Mode, using both Emphasized (vertical thickness) and Double-strike (horizontal thickness) leads to truly impressive results. So I was determined to get the printer to do the same thing in NLQ Mode. I succeeded! And the result proved to have other benefits as well! Read on...

What you must do is this: it sounds daft, but it works! Select Condensed Mode and then select Expanded Mode as well! This gives the following result, like we saw before:

Condensed AND Expanded text!

You'd think that the two modes cancel each other out, but not quite! The result should look just like the normal text size you get with standard NLQ letters. If you turn on the computer, and select NLQ, the text looks like this:

Normal NLQ text.

But if you compare this to the Condensed and Expanded text, you can see the difference:

1. This is not the same.

2. This is not the same.

The standard NLQ text (1.) isn't thickened, but the condensed and expanded text (2.) is! So you've succeeded in getting an Emphasized effect in NLQ, but that's only half the story. If you can Emphasize AND Double-strike your Draft letters, then now it should be possible to use the trick above as a method of Emphasizing NLQ text, and then select Double-strike as well, to get ultra-thick letters! And you can do just that, but it's more than just a way of thickening up your NLQ - it gives you access to a whole new range of unexpected NLQ fonts!

What happens is that the Star condenses the text, but when it expands it back to normal, somehow the vertical lines are thickened. When you then select Double-strike, the horizontal lines are now thickened, but the vertical lines corrupt this effect to the degree that the Star produced a whole new design! So instead of the normal NLQ font, you see a sort of compromise of this, which looks like so:

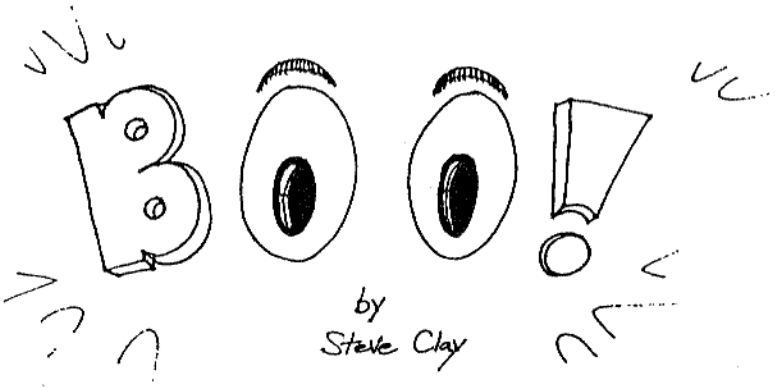
This is FONT No.1 - 'Courier'

This is FONT No.2 - 'Bansserif'

THIS IS FONT No.3 - 'ORATOR 1'

This is FONT No.4 - 'Orator 2'

Next time I'll explain another trick the Star has - you can design your own NLQ!



by
Steve Clay

Be it ghosts or chainsaws,the realm of horror and the supernatural have been one of the most popular fiction devices of all time. Why then, have there not been many good horror adventures? Is it the medium? (No pun intended)

Anyone contemplating writing a horror-based adventure has many problems to overcome! In these days of serial killers and child-abuse that the six' o clock news throws up on a daily basis, it must be extremely difficult for an author to come up with something spooky.

Keeping the theme going could also be difficult due to the structure of adventures. The need for interaction will cause pauses in the flow of the narrative. Unless a time-limit is placed upon the player,(Most players dislike this), then he/she can wait for eternity whilst that evil demon is in mid-lunge. One way around this is to place a move limit on the solving of puzzles, i.e Five moves to seal the casket before something nasty turns up.

This article evolved from my own attempts at a horror -based game. The parts that seemed to work best were those that played on things the player could accept as frightening. A demon with razor-sharp teeth isn't frightening,it is cute. (Have you seen Gremlins?) If you put the player in a chair with arms strapped down and a twisted maniac drilling into their teeth via a Black and Decker then you may hit a nerve!

Over-use of blood and gore will most likely cause apathy on the players behalf. The occasional hot-poker to the eyeball may turn the odd stomach.

The puzzles in a horror-based adventure will not be, in most cases, the focal point for creating the spooky atmosphere of a game. However, a degree of tension can be created quite easily. Example; There is room containing five caskets. As the player enters he/she is told 'You hear something...' When the player then enters LISTEN the message received could be 'You can't hear it now! It sounded as if the noise came from one of the caskets!'. Inside one of the caskets is something nasty. Putting a random element into it will keep the player and author on their toes!

I found a need for RPG-type statistics in my horror attempt. Such as heart-rate or sanity factor (How spooked are you?). You could make the heart-rate increase to the point of cardiac arrest or collapse, with the player waking up in a dentists chair.

My adventure is currently one of life's unfinished projects and looks set to remain that way for now. If you wish to tryyour hand at a horror-based game, remember that you have a vast source of background material available.

How to program a thingy (in BASIC)

By R.Tetley 1992

Everybody hates BASIC. It's slow and unfriendly etc. But is it really?, I am going to demonstrate, in a series of articles (I hope!), just how good BASIC is, I will show you some BASIC routines, which when slotted together make a finished adventure routine with the following features:

Full, advanced parser for four-word input.
Proper OOPS, AGAIN, VERBOSE, BRIEF, ALL facility.
RAMSAVE and LOAD features.
Top-screen bar, displaying score, turns and roomname.
Full wordwrap facility.
Expert handling of darkened rooms (if lightsource carried or in room, etc.)
Exit printing which properly defines commas, AND and fullstops (eg. with a n,s,nw exit, it would print up "Exits:- North, South and Northwest.")
Proper object handling, with objects' own nouns, weights and containers etc.
Decent input module, which is both responsive and pleasant to use. (NOT the crappy INPUT command!)
Full error message handling (Mistake repeating, Missing verb etc.)
Expert support routines, that can tell with one command, whether an object's noun has been typed, or whether a non-object's noun has been typed.
Machine code verb scanner, which puts the entire system up to machine code speed (it takes a fraction of a second to check 65536 (the whole memory!) bytes for a verb matchup).
The system, with 89 verbs (plus their support routines) and humourous system messages (eg. "Where is the air" illicits the response "with six-feet of your head, hopefully." along with many, many more) built into it still leaves a whopping 23K for your adventure!
Commands to print the game's vocabulary and every object's noun in brackets whenever the object is mentioned.
Intelligent parser that can strip away unwanted punctuation and convert caps into lower case.
No limit on the amount of objects, flags and locations, the only limit is the computer's memory, and that can be monitored at all times and I have recently added a feature to multi-load anything the author desires, without pain.

Too good to be true?, wrong, and anyone can program BASIC, this system is so easy it's untrue.

The Machine Code...

Seeing as this is the most important part of the program, and the system wouldn't function without it, I'll start with it.

The code itself is very small (for what it does) and it sits in the printer buffer, occupying addresses 23296 to 23364, and using address 23546 to 23551 to pass useful information.

This is the assembly listing for it, complete with comments.

```
                        ORG 23296       ; Obviously, places
                                        ; code at address
                                        ; 23296 (128K users
                                        ; shouldn't have too
                                        ; many problems re-
                                        ; locating the code)
                        PUSH AF         ; register saving
                        PUSH DE         ; bit. (on exit, BC
                        PUSH HL         ; holds verb no.)
                        LD HL,(23550)   ; address holding
                                        ; address of verbs
                        LD DE,23546     ; start of input
CHECK:                  LD B,4          ; loopcount of 4
                        LD C,0          ; number of matching
                                        ; letters found (4
                                        ; is a total match)
LOOP:                   LDA,(DE)
                        CP (HL)         ; does the data at
                                        ; DE & HL match?
                        CALL Z,IC       ; if so, call IC
                                        ; which increments
                                        ; the C register.
                        CALL COUNT      ; increment HL & DE
                        DJNZ LOOP       ; loopcounted yet?
                        LD A,4
                        CP C            ; match found?
                        JR Z,FOUND      ; branch if yes.
                        LD DE,23546     ; No, so restore
                                        ; DE back to
                                        ; start of input.
                        LD A,FFH
                        CP H            ; is H 255?
                        JR NZ,CHECK     ; if not, recheck
                        CP L            ; if both H and L
                                        ; have reached 255
                                        ; then HL=65535
                                        ; and the routine
                                        ; must end (or
                                        ; the machine will
                                        ; crash!)
                        JR NZ,CHECK
                        LD BC,0         ; no match found,

                                        ; so return a zero.
                        JR RESTORE      ; back to BASIC.
IC:                     INC C           ; increment C
                        RET
COUNT:                  INC HL          ; is it worth a
                        INC DE          ; comment?
                        RET
FOUND:                  LD BC,(23550)   ; put start address
                                        ; into BC.
                        SBC HL,BC       ; subtract it from
                                        ; HL, to get the
                                        ; character in v$
                                        ; where a match was
                                        ; found.
                        LD B,H
```

```
                    LD C,L          ; put this figure
                                    ; into BC, so
                                    ; BASIC will put
                                    ; it into a
                                    ; variable.
RESTORE:            POP HL          ; restore registers
                    POP DE
                    POP AF          ; Note the order
                                    ; Registers must
                                    ; be POPped off
                                    ; the stack in the
                                    ; REVERSE order
                                    ; to which they
                                    ; were PUSHed onto
                                    ; it.
                    RET             ; Back to BASIC.
```

The code itself has a lot of demands before it can be used, so that it doesn't crash the computer...

The verbs must be stored so that they are all in one continuous string, containing the first four letters of each verb and the line number in BASIC they are situated at, eg..

```
10 LET V$="INVE0900SAVE3000LOAD2355HELP2345GET
0400TAKE0400.."
```

(use spaces when entering verbs with less than four letters, to make them four letters)
This string must then be poked into memory, (anywhere will do) I would suggest that you use the last address as 65367 as anything higher than this can have some unusual (and often fatal) effects on the machine, and it leaves you the UDG's free for input prompt icons etc.

It is important that the last address used to hold a value of the string (in my example 65367) when subtracted from 65535 is divisible by 8. In my example, 65535 - 65367 = 168, 168 is divisible by 8, so the machine won't crash.
This is a short program to poke the string into memory:

```
10 FOR F=1 TO LEN V$: POKE (65367-LEN Z$+F)+1,CODE V$(F):
   NEXT F
```

You must then poke the value of the first address into memory locations 23550 (low) and 23551 (high), to find this address, subtract LEN V$ from your top address (in my case 65367), here is another short program to do it.

```
10 LET V=65367-LEN V$:LET N=23550: POKE N,V-256*INT
(V/256):POKE N+1,INT (V/256)
```

Then (phew!) you must poke the four letters of the player's input into addresses 23546-23549, here's another program (pretending the input is in a$).

```
10 FOR F=1 TO 4: POKE 23545+F,CODE A$(F):NEXT F
```

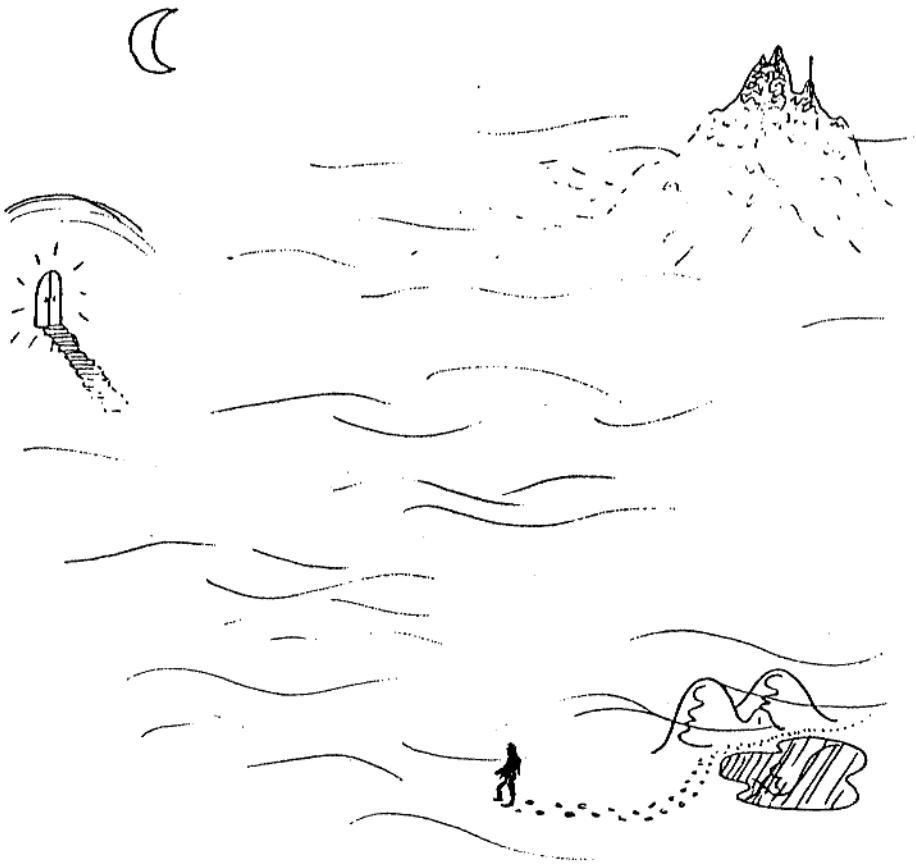
Once you have set these parameters, call it with:

LET VERB=USR 23296, and VERB will be equal to the character in v$ where a match was found, to then go to that verb, simply use GOTO VAL V$(VERB+1 TO VERB+4), what could be easier ? (lighting a match on soap, perhaps?)

Also, a quick note, it is a sensible idea, once you have poked 23550 and 23551 with the address of the verb start to copy these two values into 23730 and 23731 (the RAMTOP address), this way you protect the verbs from being overwritten by BASIC.

I hope that this routine is of use to people, and I shall hopefully (if I get time!) be sending in my BASIC system. (My problem is that I don't own a printer or typewriter, so I have had to do this on Wordstar on the IBM 386/N and Xerox laser printer at work (isn't life hell?!))

Hope this piece of code is useful to some people who swear by BASIC.



THE TAXMAN CHRONICLES

by Steve Clay

This article is Part 1 of a series charting the rise and fall and rise again of an adventure. The game in question is the sequel to 'The Taxman Cometh' and is aptly titled 'Tax Returns'. For those who don't know, the aim of the Taxman games is to collect unpaid taxes from various fantasy-type characters who put all sorts of obstacles in the player's way, of course.

Looking back, (aren't we all wiser in hindsight?) I can see where I went wrong intitially with Tax Returns. Encouraged by favourable remarks from various playtesters, I rushed through the planning stages of the game and linked together a couple of decent puzzles with a string of seriously dodgy ones. I had the map and details sorted out in two nights! Considering the first game had taken months of preparation I should have realised that trouble was over the horizon.

I use PAW and I had to to program nearly 14K of game before the poor quality of the puzzles became apparent to me. I had been through much the same thing with the first version and I look at it now as a creating phase and useful, if only to get rid of some grotty ideas! Version 1 was dumped without any tears. My principle is if I don't enjoy writing the game then nobody is likely to enjoy playing it.

The original version contained characters from fairytales 'Rapunzel', 'The 3 Pigs', etc. At the time of writing these are being incorporated into a third Taxman game. From here I came up with Snow White and the seven dwarves, eight debtors to play around with. What's more I could play around with the names of the dwarves for my own purposes. My initial seven were Boozy, Loner, Fixer, Eyecue, Gadget, Caster and Trapper. Along with Snow White they would live in and around the diamond mine where they made their fortunes.

To avoid numerous locations that described themselves as a passage or a corridor or a cave I decided to join the various levels of the mine with a lift room. Three of the dwarves and Snow White would reside outside the mine to avoid the said repetition. The player starts outside the mine and to reach the room with the lift in (I named this the LIFT ROOM, what cunning) I thought it would be nice to give the player a nice introductory puzzle. Let's see. Light! The old chestnut. Not just a simple lamp oh no! What I have devised is a 3*3 grid of rooms, each changing colour when the player enters it. These are the light generation rooms. There is a corridor that skirts the western edge of the rooms, this is to allow access;

R1 - R2 - R3 - Corridor
R4 - R5 - R6 - " (R denotes Room)
R7 - R8 - R9 - "

At the northern end of the corridor is an LCD panel. LCDs being Light Carrying Dweevils, magical creatures that change colour between red and green dependant on what impulse they receive. The LCDs are also set out in a 3*3 grid. When the play starts the LCDs are all red. The idea being that the player should enter each room just once to turn the LCDs green. It can be done other ways but this is for wimps.

To program this section was fairly straightforward. As I mentioned before I use PAW. I have a system when writing which leaves Location 0 as the title page, Location 1 is a bin (vital to have a dumping room because LOC 252 can cause

all kinds of trouble if used without care and thought), Locations 2 to 9 are all reserved for containers. So my first game location is normally 10. Going north from here puts the player in the light room sector, the southern end of the corridor to be precise. So what does PAW do here?

Process 1 checks if the player is within the light room area, if it is it jumps to Process 6. (Process 3 is used to print the exits, 4 and 5 are used to tell the player who has and hasn't paid.) Process 6 then loads the flag of the current room into a working flag and calls Process 7 where the contents of the flag are altered. An easier way to explain is to show extracts from the database:

```
PROCESS 1
*   *   ATGT 14 ATLT 24 PROCESS 6 ; Locations 15-23 are the
PROCESS 6                           light rooms
*   *   AT 15 COPYFF 60 69 PROCESS 7 COPYFF 69 60
*   *   AT 16 COPYFF 61 69 PROCESS 7 COPYFF 69 61 ; A similar
entry for each room until...
*   *   AT 23 COPYFF 68 69 PROCESS 7 COPYFF 69 68
*   _   NOTZERO 69 MES 0 ; MES 0 = "GREEN"
*   _   ZERO 69 MES 1    ; MES 1 = "RED"
PROCESS 7
*   *   ZERO 69 SET 69 DONE
*   *   CLEAR 69
```

A bug quickly came to light. If the player entered the light rooms he/she could alternate the colour by simply (R)edescribing the location. The simple solution involved putting a flag on the R command:

```
RESPONSE
R   _   ATGT 14 ATLT 24 SET 70 DESC ; then...
PROCESS 7
*   *   0 NOTZERO 70 CLEAR 70 DONE ; If the player was
redescribing the location the room colour would not change.
The CLEAR 70 command is also vital; if it is left out the
player would not change the colour of the room even when
entered legally.
```

The basics of this helped when the player examined the panel. I used Process 8 and loaded the flag for each room into the working flag. The relevant message was printed:

```
RESPONSE
X PANEL AT 14 MES 2 COPYFF 60 69 PROCESS 8 COPYFF 61 69
PROCESS 8 COPYFF 62 69 PROCESS 8 NEWLINE etc. The NEWLINE
```

appeared after every 3 commands to give a result like:

```
The panel is (etc). Its current state is
RRG
GRG
RRR
```

Process 8 deals with which message to print:

```
PROCESS 8
*   *   ZERO 69 MES 4 DONE ; MES 4 = "R"
*   *   MES 5              ; MES 5 = "G"
```

So that is the light rooms up and running, next time the lift room and beyond!

The State Of The Adventure Scene Today

by Ian Eveleigh

Here we go... the state of the adventure scene today... time to harp on about ''the good old days'' when a Speccy game used to be under 6-quid (and you'd still have change for fish & chips on the way home.) Well, er, no actually! Let's try a new approach. Now, heck, call me radical if you like, but I'm going to suggest that (get ready to duck) the adventure scene is, in fact, DEAD! Yup, a real gonner! Kicked the bucket. A corpse. A stiff. (I bet Chris puts 'OO-ER' in brackets here...) *(Chris - actually, I was going to put 'An EX-adventure scene! But when I purchased this adventure scene from this establishment not half an hour ago, you assured me it was just resting!')*

Come on, face it. When was the last time you went into a computer shop and bought an adventure? No, not including those puffy animated/menu driven jobs. I mean a MAN'S adventure! Think back... Bet it was a Magnetic Scrolls game. (Or a Level 9 one if you're a real tight git!) And how many games have you bought since then? One? (Ooo! Someone just said ''No, two ACTUALLY.'' Well done!) Bet it was a Zenobi game? Thought so.

So let's just re-assess the adventure scene shall we? Right. There's Magnetic Scrolls, and there's Zenobi, and there's bugger all else. Mmm. Ok, maybe DEAD was an over-reaction - more sort of near-dead, twitching with its last grasp on life.

The problem is adventures are getting pushed out because they're just not money-makers anymore. *(Because any one can now produce one? - Ed)* Kids want the thrills and spills of 'Sonic The Hedgehog' or the cute and colourful challenge of 'Lemmings'. They don't want a BOOK! That's where the money's at. Why buy a 400 quid computer to look at text when you could have mega impressive graphics displays and attention grabbing action?

Thus adventures have been left to a small group of die-hards. Magnetic Scrolls keeping up the on-the-shelf front (but only just, and even they have had to edge towards WIMP; but more tactfully done than most.) and then there's good old Zenobi (but then I would say that, wouldn't I?) churning out some good old purist gear.

There is one other problem, I'm a victim of this one myself - 8-bits are just proving too small to write effective, Nineties adventures on. Programmers really need the space of a 16-bit, but, Sod's Law, there is a lack of quality writing utilities on the 16-bit machines (especially ones that are on more than one machine). This is a real dead area - it appears Gilsoft have abandoned us, and there's no real hope on the horizon. Personally I'd love to code my (already developed) second adventure, but I just cannot find a suitable and flexible enough 16-bit utility, and after becoming accustomed to the slickness of my own PC and some very hot UNIX based machines just over from the U.S. that I use for many hours a week, there is NO WAY I could go back to the sheer stubbornness of an 8-bit. I guess we just have to hope for a miracle in this area. Any suggestions?

Let's face it, adventuring has been reduced to a very select hobby, and it's now up to US to keep it alive. Don't moan about big companies not releasing adventures - would you? No. So it's time to look to the future - it's time to support the smaller software houses, support magazines such as 'Probe' and 'Coder' - even contribute! C'mon, let's give this corpse some shocks...

PRIDE OF THE FEDERATION

Reviewed by George E. March

Just recently I was sent a copy of a game, by Chris, that I was asked to 'review' with an eye to this game being given away free as a gift when buying Adventure Coder. 'Great!' I thought, 'Just what we need to please our readership, just like the big magazines, ZZAP!, Your Sinclair, etc, etc, have!' But...

'Pride Of The Federation' by Excalibur Software (anybody heard of them before? I haven't!) was Copyright 1987, so presumably they tried to release it then, but didn't have much (if any?) luck in getting it sold.

Why didn't Pride sell many copies the first time it was released? It *WAS* written with PAW after all. A very decent utility, allowing graphics, various modes (like still and scrolling text), fonts, etc. So what happened?

Even though Pride was written with PAW, as I said, and PAW *does* do graphics, Pride does NOT have any graphics, BUT this is not what I'm bothered about. I prefer text-only games after all! What does bother me though are the funny little inconsistencies in the programming routines! Why, if Pride does not have any graphics, does is still include memory wasting GRAPHICS ON/OFF routines? There really hasn't been any attempt at all to try and, as they say, 'tart it up a bit' with any new routines. For example, maybe an AGAIN routine could have helped, or lengthier text, whatever. So basically, Pride, using only the standard built-in routines, simply 'feels' like your average, 'bog-standard' medium-level PAWed text game.

Well, enough of the gripes (for now!). The game is actually set on a farm (but not a cows and pigs kind of farm, you understand!) aboard a space-station (though you wouldn't know it) that grows legalised poppies for their opium, to be used in customised drugs! The idea behind the game is different I must admit!

This station is run by robots, and you seem to be the only human on board, even though the game does say that there's someone else there, but I couldn't find him/her.

When you actually get into the game, you play the character of John Pride...

QUESTION: Why is it John and not Joanna? Why are there hardly any female characters used in adventure games, or any games at all for that matter? I know Adventure Probe's co-founder Sandra Sharkey (who wrote the GAC/STAC versions of 'Shymer') uses females as lead characters, as does Probe co-founder Pat Winstanley, when she wrote 'Toil & Trouble' for the ST, as does ex-Probe editor Mandy Rodrigues, and Marlin Games' own Linda Wright (who's written many, many of my all-time favourite games, text-based and graphics. I thoroughly recommend you pick up one or two of her titles, now available from Zenobi Software), and Ruth Sunderland (who's now getting her games published through PD).

Now, the game itself has four loads, POF1 - POF4, with game-saves inbetween each, and the quite decent inlay card lists enough of the 'standard' inputs to get even a complete novice started.

When you do start the game (first load) you're presented

with such an awful (pale-red text on a dark-red background) title page that I was nearly tempted to beat the crap out of the TV I was using! Thinking it was the TV at fault, it wasn't! As I found out when the actual game started.

The very short description of each location (which are your usual 'You are sitting... You are standing...' etc) is also in these same god-awful (pink on red) colours. The available exits are listed in dark-blue on pale-blue, and are printed right underneath the location descriptions. It's a pity there wasn't a blank line inbetween the two, it might have looked a bit better! But, weirdly enough, any messages printed, inputs by the player and the prompts themselves are all printed in easily readable yellow!

When I attempted some of the usual playtester inputs, like examining or searching things that ARE in the room description, I again received some mediocre replies...

I EXAMined the TREE in the first location (you're up a tree at the start of the game) and got 'You can't examine that' printed. (Hmm?)

I SEARCHed the same TREE and got 'I was not able to understand any of that. Please try again'. (Ah-Hem!) Yes, well, I then examined/searched some things that weren't in the current room description... and got exactly the same answers as I had above (Well, well, well?). I should at least have got a decent reply for objects that *were* there!

The HELP command is of no 'real' help whatsoever (all it gives is an address to write to and a phone number!) *(Ed - the author of the game tells me not to write or phone to that address, as it is no longer relevant. If you want any help on Pride, you can now write in to Coder, as I have the complete help sheets - all thirteen pages of 'em!)*

The messages in the Systems Messages Table are all pretty standard with little (if any) action taken by the author to try and get either the sys' messages or Response routines to look or act any better than they were originally, but...

When it all comes down to the actual playing of the game, it's not at all bad, really! With quite a few really hair-tearing problems to cure, especially trying to figure out which switch to turn off the scanners with the spiders watching (Oooh! Nasty!), and it is a free gift after all! I think my main mistake was that I was actually playtesting it and not just playing it, so I'm possibly being a bit too harsh on it, but what do you think?

How to Make a Thingy (Part III)

by Stephen Groves

Right then, time to take a break from typing for a few minutes. Don't try to assemble what we've done, because it won't work ....yet.

Before we start on the next section, I'll explain a bit more about the tables.

First of all, I'll list all the table names, memory locations and lengths and then I'll describe what they're used for.

TABLE NAME	STARTS AT	LENGTH
MAINOB	24785 (60D1H)	100 bytes
MAINEX	24885 (6135H)	150 bytes
MAINFL	25035 (61CBH)	255 bytes
OBJPOS	25290 (62CAH)	100 bytes
EXITS	25390 (632EH)	150 bytes
FLAGS	25540 (63C4H)	255 bytes
OBJTAB(1)	25795 (64C3H)	100 bytes
OBJTAB(2)	25895 (6527H)	100 bytes
LOCTB1	25995 (658BH)	150 bytes
LOCTB2	26145 (6621H)	150 bytes
NTAB	26295 (66B7H)	150 bytes
NETAB	26445 (674DH)	150 bytes
ETAB	26595 (67E3H)	150 bytes
SETAB	26745 (6879H)	150 bytes
STAB	26895 (690FH)	150 bytes
SWTAB	27045 (69A5H)	150 bytes
WTAB	27195 (6A3BH)	150 bytes
NWTAB	27345 (6AD1H)	150 bytes
RAMBUF	27495 (6B67H)	505 bytes

MAINOB

Each object in the adventure is given an object number. For example, a sword might be object No. 1, a shield object No. 2 and a ring object No. 3. Although the input routine will recognize the words sword, shield and ring, it is easier and quicker to manipulate numbers. So although the player will want to drop the sword, the computer will drop No. 1.

The MAINOB table has one byte for every object, and we have left enough space for 100 objects. This should be more than enough for most adventures, but can be expanded to 255 if necessary.

So, each object has its own memory location. The sword's memory location is at MAINOB +1, the shield at MAINOB +2 etc etc. To manipulate the object, we need to know where it is at any given moment. We do this by putting the physical location of the object into it's memory location.

An example :

Adventure location number 14 is a field of stubble on the edge of a forest. The sword is on the ground at this location.
Adventure location 20 is the deserted cellar of a tavern. The shield is on the wall.

MAINOB +1 (the sword) contains the number 14.
MAINOB +2 (the shield) contains the number 20.

If the object memory location contains zero, then the object has not yet been created. If it contains 255 (FFH) then the object is carried and if it contains 254 (FEH) then that object is worn. Any other number (1 to 253) will show the location in the adventure where that object is. MAINOB is the master object table. It's contents are copied to OBJPOS where all tests and manipulations take place.

MAINEX

MAINEX is the exit table. As explained earlier, this is used to show what exits exist from all the adventure locations. For example, if location 14 above has exits to the NORTh, SOUTHEAST and WEST, then MAINEX +14 would contain 10010010 in binary or 146 decimal. This table is the master exits table, it's contents are copied into EXITS which is the working copy where all tests and manipulations take place.

MAINFL

MAINFL is the master flags table. It's entries are copied into FLAGS where all tests and manipulations take place.

As you will have noticed, the first three tables are all master tables and as such their contents are not altered during the course of the adventure. This then allows us to restart the adventure at any time, and by copying the contents of these tables to their 'working' counterparts, the adventure will start in the same place every time.

OBJPOS

The working copy of MAINOB. This table is altered throughout the adventure as objects are moved, created or destroyed. OBJPOS is initialised every time the adventure is restarted.

EXITS

The working copy of MAINEX. This table is altered whenever a new exit is created. For example if a door is unlocked or a wall demolished, a new exit is created and EXITS would be amended accordingly.

FLAGS

The working copy of MAINFL and possibly the most important table of all. FLAGS is accessed using the IX regigister pair and an offset number eg. LD A,(IX+0AH). Because the offset is calculated as being in the range -128 to +127, IX actually points to 25668 (6444H) and not at the beginning of the table (25540). However, all references to flags will show the offset between 0-255.

I have designated all flags below 32d (20H) as being for the sole use of the system. Not all have been given uses yet, but it is always better to be on the safe side. All flags above 32d will be available for your own use.

An explanation of each flag will be given as we get to it, but here are some of the main ones with their uses.

FLAG No.	CONTENTS/USE
0 (00H)	0=location light. 1=location dark.
1 (01H)	0=lamp off. 1=lamp lit.
10 (0AH)	Present location No.
11 (0BH)	Current verb No.
12 (0CH)	Current noun1 No.
13 (0DH)	Current preposition No.
14 (0EH)	Current adjective1 No.
15 (0FH)	Current noun2 No.
16 (10H)	Current adjective2 No.
17 (11H)	Max No. of items carryable.
18 (12H)	Number of items carried.
25 (19H)	Graphics main flag.
27 (1BH)	Speech flag.
28 (1CH)	Duplicate noun for pronoun.
30 (1EH)	Duplicate verb.

OBJTAB(1) and OBJTAB(2)

Each object used in the adventure has a description. This description is used either when the object is first seen, or when it is examined. Obviously this description is held somewhere in memory, and the start address of each description will be a 16 bit number. To tell the printing routine where each description starts, I have split OBJTAB into two tables. Each table is 100 bytes in length, and OBJTAB(2) follows directly from OBJTAB(1). The low byte of the description address is placed in OBJTAB(1) and the high byte in OBJTAB(2). In both cases, the address byte is placed in the position of OBJTAB(1)+object No. and OBJTAB(2)+object No. Another brief example shows how it works:

Load HL with the address OBJTAB
Load DE with the object number (say 'the ring' =3)
Add HL and DE, leaving HL containing OBJTAB+3
Load E with the contents of (HL), the low byte of object description.
Load BC with 100d. The difference between OBJTAB(1)+3 and

OBJTAB(2)+3
Add HL and BC. HL now points at OBJTAB(2)+3.
Load D with the contents of (HL).
DE now contains the address of the description of object 3.

LOCTB1 and LOCTB2

As with OBJTAB, each location description is stored in memory with a 16 bit number showing the start address. This number is again stored in LOCTB1 and LOCTB2. The low byte is held in LOCTB1 and the high byte in LOCTB2. A similar routine to the one used above, constructs the address of the location text, before it is passed to the printing routine.

NTAB NETAB ETAB SETAB STAB SWTAB WTAB NWTAB

During the course of the adventure, if a player types in a direction, the programme first checks with the EXITS table to ensure that and exit actually exists in that direction. The exit word is then converted to a number, 1 for NORTH to 8 for NORTH-WEST. The correct direction table is then found and the current location number is added to the start of that direction table. The number that is stored at the resulting memory location within the direction table is the new location number that is moved to by travelling in that direction.

RAMBUF

The final table is called RAMBUF, and as may seem obvious, is the buffer used during the game for the commands RAMSAVE and RAMLOAD. This table is 505 bytes long, and when used, contains the contents of OBJPOS, EXITS and FLAGS. It can be seen, that by saving just these three tables, a previous position can be recalled at any time.

Now that we have an understanding of the tables, we can carry on with the programme. We continue with SMINV which prints the small inventory in the top part of the screen when in graphics mode.

```
;Registers A, BC, DE and HL are destroyed in this routine.

OBJPOS          EQU 62CAH       ;The beginning of the
                                 OBJPOS table
SMINV           LD BC,(COLPOS)  ;By loading COLPOS, we
                                 place COLPOS in B and
                                 LINPOS in C. BC now
                                 contains the next print
                                 position.
                PUSH BC         ;Save this position on the
                                 stack.
```

```
                LD BC,0110H     ;The position for SMINV
                                 printing (LINPOS is
                                 actually LINPOS -1 as it
                                 is incremented before
                                 printing)
                LD (COLPOS),BC  ;The position is now
                                 stored in COLPOS/LINPOS
                                 for printing
                LD B,64H        ;Place the length of the
                                 object table in B for use
                                 as a counter
                LD HL,OBJPOS    ;The start of the object
                                 table
SMINV1          LD A,(HL)       ;Load A with the position
                                 of object No 1
                CP 0FEH         ;Test to see if it is
                                 carried or worn
                JP NC,GOT1      ;If it is, then jump to
                                 GOT1
                INC HL          ;If it isn't, increase HL
                                 to the next object
                DJNZ SMINV1     ;Loop back if B>0 (more
                                 objects to be tested
SMEND           LD BC,(COLPOS)  ;Load the print position
                                 into BC
                LD A,8          ;Load 8 into A and....

                CP B            ;..compare to see if 8
                                 lines have been printed
                JP Z,SMEND1     ;Jump to SMEND1 if they
                                 have
                INC B           ;If not then print on the
                                 next line
                LD C,10H        ;The column printing pos'n


                LD (COLPOS),BC  ;Put the new print pos'n
                                 back
                LD BC,TENSPC    ;Load BC with the location
                                 of 10 blank spaces
                LD (LOCPOS),BC  ;Place this in LOCPOS for
                                 the printing routine
                CALL LOCPRN     ;Print the 10 spaces

                JP SMEND        ;Return for the next line

SMEND1          POP BC          ;When 8 lines have been
                                 printed, restore the
                                 original print pos'n from
                                 the stack
                LD (COLPOS),BC  ;Return it to COLPOS...

                RET             ;...and return from SMINV

GOT1            PUSH HL         ;Save the position in
                                 OBJPOS table to the stack
                PUSH BC         ;Save the counter

                CALL TADGET     ;A routine that returns DE
                                 pointing at the object
                                 name
```

```
                EX HL,DE        ;Transfer the address to
                                 HL
                LD DE,(COLPOS)  ;Take the previous print
                                 pos'n...
                INC D           ;..and increase to the
                                 next line...
                LD E,10H        ;.. and new column No.

                LD (COLPOS),DE  ;Store it for LOCPRN

                CALL PLOCR      ;A small routine that
                                 saves a few bytes every
                                 time. It makes a call to
                                 LOCPRN
                POP BC          ;Restore the remaining
                                 length of OBJPOS
                POP HL          ;Restore the current
                                 position in OBJPOS
                INC HL          ;Move to next position
                                 in OBJPOS...
                JP SMINV1       ;..and jump to continue
                                 checking
TENSPC          DEFM "          ";Ten spaces

                DEFB 0DH        ;The terminating byte to
                                 signify to LOCPRN that
                                 the end of a section of
                                 text has been reached.
LOCPOS          DEFW 0          ;Storage position used by
                                 LOCPRN
```

Writing tips

by Steve Clay

There follows a collection of adventure writing tips from around the world;

Are your graphics letting your adventure down? Do your trees look like lollipops? Then what you need to do is to stop all this pretence and admit you can't draw. Stick to writing text-only games!

Run out of ideas? Need a puzzle? Then why not hack an old game and pinch all the puzzles! I'm sure nobody would notice!

If you make it big in adventures then relax! You can now churn out any old twaddle and no one will have the courage to tell you!

If you wish to make money from another source then why not send stupid tips in to magazines! Some give you lots of money for doing so. (Not this one we hasten to add)

And now it's time to Ask Aunt Pru! Our resident (for this issue only) agony aunt!

Dear Aunt Pru,

In my brilio adventure(what is being wroted by me!) there is a mouse who needs some cheese! How do you fit the cheese into the computer? Do you need an interface? The mouse was easy I just crammed it through the joystick port.

Yours STUCK.

P.S The mouse is starting to smell. Is this normal?

Dear Stuck,

What a pity you didn't write sooner. What you should have done was liquidize the mouse. This saves a lot of room. As for the smell, well I think it's only natural,don't you? The cheese problem is a little difficult. Why not try cottage cheese or one of those smelly french cheeses? Anyway whatever you decide keep at it, your game sounds a real winner.

Disclaimer: Following Aunt Pru's advice would be an immensely stupid thing to do!

ADVENTURE CODER 19 *Version 1, November 92*

SUBSCRIPTIONS: get yer Coders here! The price per single issue/back issue if you live in Britain remains £1.25 despite the recession! (Other mags dare to charge £2!!) Elsewhere costs £1.42 at the slow Seamail rate, or £1.47 Airmail to Europe! If you live in a Zone 1 area (Canada, Egypt, The Falklands, Saudi Arabia, The Seychelles, USA, etc) then the price is £1.74 Airmail. For Zone 2 dwellers (Australia, New Zealand, etc) your price is £1.83 Airmail.

PAYMENT: By cheque, Postal Order, or International Giro, all in Sterling (for what it's worth) please. British decimal stamps accepted for small amounts, but NO coins! (Especially not those new 10p ones!!) * CHEQUES: Payable to "C.HESTER" only.

ADVERTISING: Inside page: £5 * Back cover: £6 * Half page (inside): £2.50 * Quarter page (inside): £1.25. Line ads: FREE. All advertisers get a free issue with their advert in, except when line ads only are used.

CONTRIBUTIONS: Anything welcome - cartoons, drawings, articles, letters, program routines, tips... Printed articles should be in dark ink with a margin all round of about this width: <------>. Articles printed out with the text going to the edge of the sheet mean I have to type them up, causing much aggravation (March and Clay!). It saves evenings of work if stuff is printed with a margin!! (So you'll get the next magazine sooner!) But handwritten articles are also welcome! Some companies now refuse to accept these. Imagine a book publisher turning down a set of handwritten plays by a new playwright, W. Shakespeare! Printed text is no measure of the quality! Handwriting is no crime! Yet now every new writer has to have a word processor - sickening, eh? What if they're the next genius, but can't afford a computer, and write with a biro?

You can also send text on a disk if you like, so long as it's a 3½" one, formatted to MS-DOS standard (as used by PCs, Amigas and STEs, etc.) And your text must be saved as ASCII text - that is, with no printer control codes in it! That way I can print the text myself. (All disks returned, of course!)

EDITOR: C.Hester * LAYOUT: Editor * DISTRIBUTION: Layout Artist * FORMAT: Amateur non-profit magazine * FOUNDED: July 1989 * ADVENTURE CODER is produced using ST Writer Elite 4.2 and Write On, running on a STEctrum, with 16 colours (out of 4,096!) and 1 channel sound (times 8!) with 48K (plus 952K!). Printed using a BJ-10ex printer (8-pin (times 8!!! Honest!)). Who needs a laser printer?

ADDRESS: All correspondence to this address only:

Christopher Hester,
3 West Lane,
Baildon,
West Yorkshire,
BD17 5HD.